SPC1169 DAC COMP PGA 使用指南

版本 A/0 – 2023 年 5 月

概述

在电力电子系统中，DAC，COMP，PGA 有时会组合起来，采用 PGA，DAC 作为 COMP 的输入，从而检测过流事件，触发 PWM 封锁保护。

目录

图片列表

表格列表

版本历史

版本	日期	作者	状态	变更
A/0	2023 年 5 月 11 日	Hang Su	Released	首次发布。

术语或缩写

术语或缩写	描述

1 DAC 实例

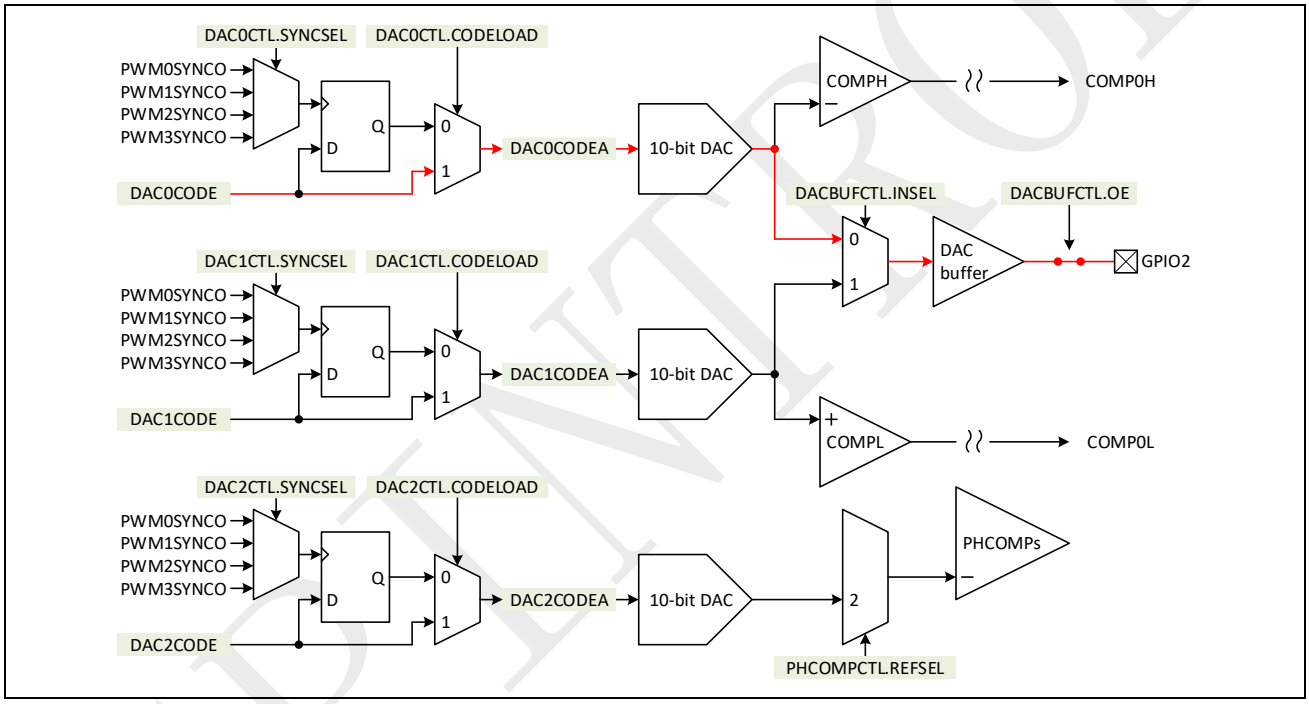
1.1 DAC 转换并输出到引脚

DAC 最常规的用法，是将 DACXCODEA 直接送入 DAC，从而产生模拟电压，如图 1-1 所示。将 DAC0CODE 的数值经 10-bit DAC 转换为模拟信号，该模拟信号既可直接传给 COMP，亦可经过 DAC buffer 直接输出到 GPIO2。

$$V_{out} = VDD3 * \frac{CODE}{1024}$$

通常 VDD3 == 3.3V。

图 1-1：DAC 示意图



以下一个例子通过 DAC 在 GPIO2 输出 1.65V 电压：

Example Code

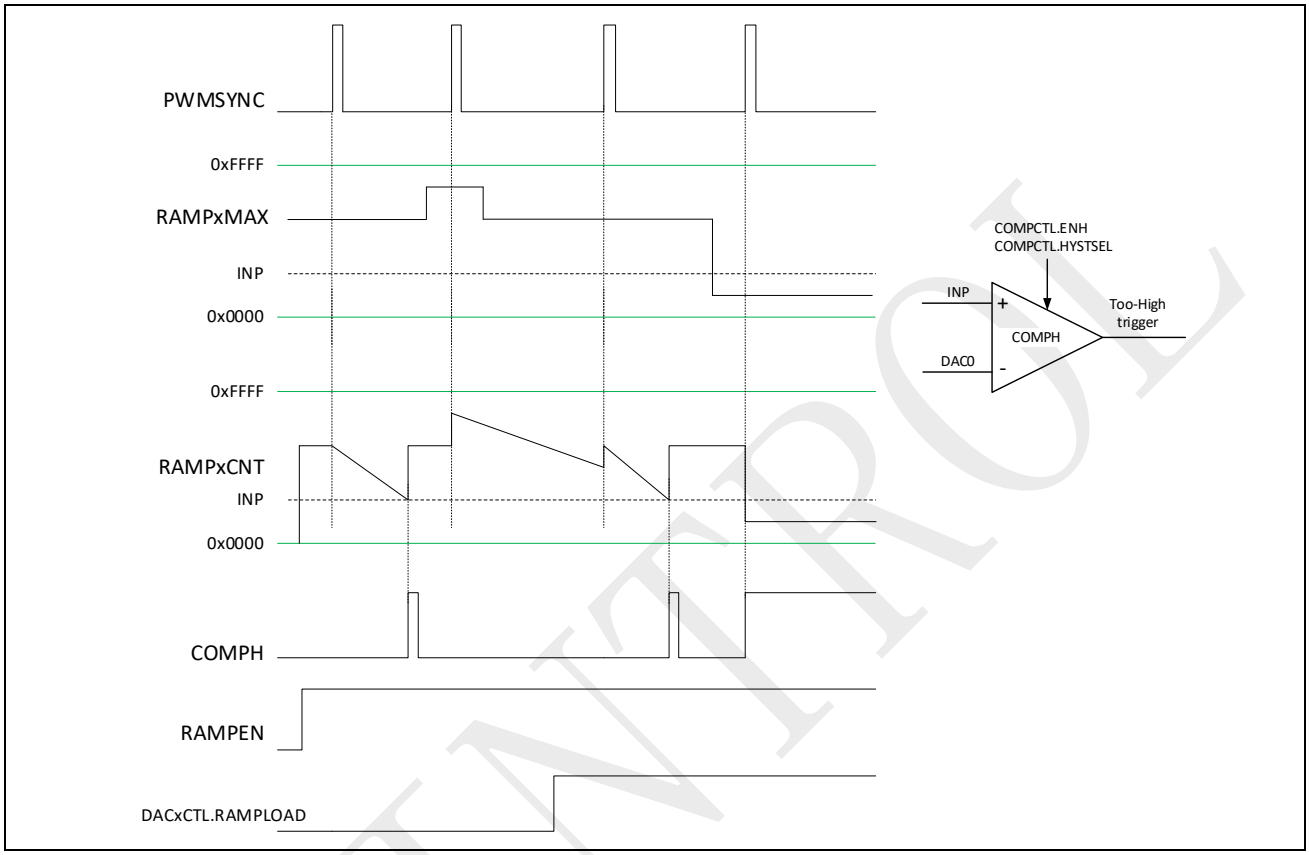
```
int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);
    printf("Enter the test\n");
```

```
    /* enable DAC0 */
    COMP_EnableDAC(DAC0);

    /* Set DAC0 as Direct mode(DAC code is immediately update) */
    COMP_SetDACCodeLoadTiming(DAC0, DIRECT_LOAD_MODE);

    /* set DAC0 output to 1.65V  */
    COMP_SetDACValue10Bit(DAC0, 512);

    /* DAC Buffer Init */
    COMP_DACBufferInit(DAC0);

    /* Enable DAC Buffer Output To GPIO */
    COMP_EnableDACBufferOutputToGPIO();

    /* Init ADC and set DAC buffer as the positive input of the ADC CH0 */
    ADC_EasyInit1(ADC, ADC_CH0, ADC_IN_DAC0, ADC_SOC_TRIGGER_FROM_SOFTWARE);

    /* Enable the global INT for the ADC */
    NVIC_EnableIRQ(ADCCH0_IRQn);

    /* Use software to trigger ADC CH0 to work */
    ADC_ForceChannelSOC(ADC, ADC_CH0);

    while (1)
    {

    }
}
```

1.2 斜坡发生器

PWM, DAC,COMPH 可以共同构成斜坡发生器，产生斜坡电压 RAMPxCNT，如图 1-2 所示。

图 1-2：斜坡发生器

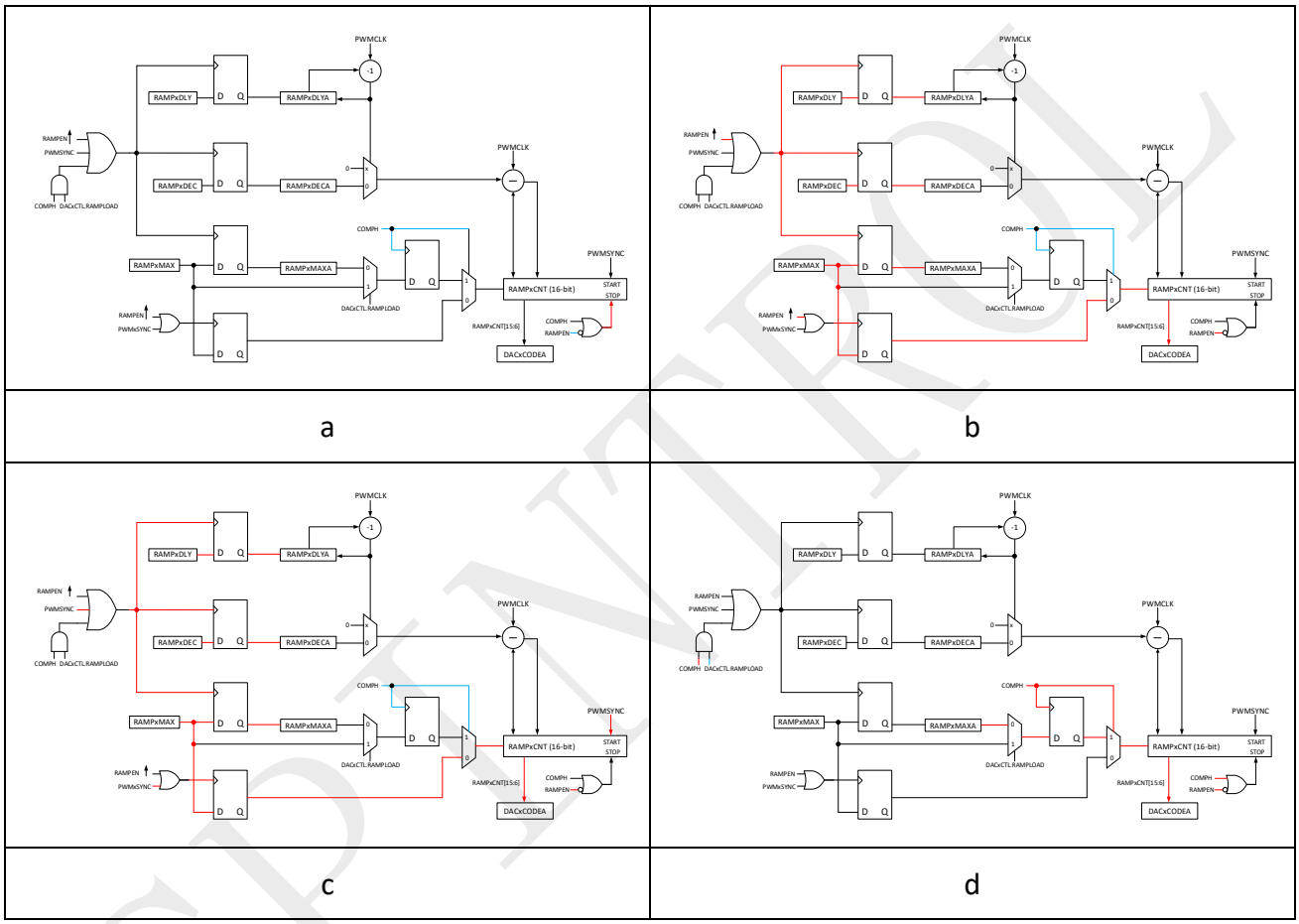


- 因为 DAC0 只能接到 COMPH 的负端，为了保证默认状态无 COMPH 事件，必须保证默认状态 DAC0CODEA(RAMP0CNT[15:6])电压 > INP；
- 初始条件 COMPH 是禁止的（通过软件设定），因此虽然 INP > DAC0CODEA(RAMP0CNT[15:6]) 电压，但是没有对应的 COMPH 事件；同时 RAMPEN 为低，停止 RAMPxCNT 的递减计数，如图 1-3a；
- 下一时刻，RAMPEN 产生上升沿脉冲，由于此时 COMPH 仍是禁止，没有 COMPH 事件，从而控制下路连通，RAMPxMAX 通过下路触发器直接加载到 RAMPxCNT，同时上路的各寄存器初始值均加载到对应的活跃寄存器；当 RAMPxCNT 加载完成后，其[15:6]位构成 DACxCODEA，在其模拟电压稳定后，使能 COMPH，此时 DAC0CODEA(RAMP0CNT[15:6])电压>INP，没有 COMPH 事件，COMPH 事件维持低电平图 1-3b；
- 下一时刻，PWMSYNCO 产生上升沿脉冲，由于没有 COMPH 事件，从而控制下路连通；RAMPxMAX 通过下路触发器直接加载到 RAMPxCNT，同时上路的各寄存器初始值均加载到对应的活跃寄存器，同时 PWMSYNCO 也会使能 RAMPxCNT 的递减计数图 1-3c；
- 当 DACxCODEA 电压递减到<INP 时，COMPH 事件触发，由于 RAMPLOAD 为 0，RAMPxMAXA 通过触发器直接加载到 RAMPxCNT，COMPH 事件消失，同时 COMPH 关闭 RAMPxCNT 的递减计数图 1-3d；

- 下一时刻，PWMSYNCO 产生上升沿脉冲，由于没有 COMPH 事件，从而控制下路连通；RAMPxMAX 通过下路触发器直接加载到 RAMPxCNT，同时上路的各寄存器初始值均加载到对应的活跃寄存器，同时 PWMSYNCO 也会使能 RAMPxCNT 的递减计数图 1-3c；

- 下一时刻，PWMSYNCO 产生上升沿脉冲，由于没有 COMPH 事件，从而控制下路连通；RAMPxMAX 通过下路触发器直接加载到 RAMPxCNT，同时上路的各寄存器初始值均加载到对应的活跃寄存器，同时 PWMSYNCO 也会使能 RAMPxCNT 的递减计数图 1-3c；

图 1-3：根据 PWM 同步输出信号递减电平



示例演示了通过定时器产生 SYNCI 信号并通过 SYNCO 触发 DAC 递减计数，从而产生斜坡电压。

实验时，COMPH 对应正端电压必须大于 0V，否则负端电压无法降落到正端电压之下。

Example Code

```
int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    Delay_Init();

    /*
     * Init the UART
     */
```

```
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);
    printf("Enter the test\n");

    /* enable DAC0 */
    COMP_EnableDAC(DAC0);

    /* Set DAC0 as Direct mode(DAC code is immediately update) */
    COMP_SetDACCodeLoadTiming(DAC0, DIRECT_LOAD_MODE);

    /* set DAC0 output to 3000mV  */
    COMP_SetDACVoltage(DAC0, 3000);

    /* DAC Buffer Init */
    COMP_DACBufferInit(DAC0);

    /* Enable DAC Buffer Output To GPIO */
    COMP_EnableDACBufferOutputToGPIO();

    /* Select the PWM synchronous output signal for DAC */
    COMP_SetDACSyncEvent(DAC0, SEL_PWM0);

    CLOCK_EnableModule(PWM_MODULE);

    /* Connect the input sync signal with output */
    PWM_SetSyncOutEvent(PWM0, SYNCO_SYNCI_AND_FRCSYNC);

    /* Enable PWM SYNC by the signal coming from TIMER1 */
    PWM_EnableSyncFromTIMER1(INC_PWM0 );

    printf("PWMCFG->TMR1SYNCIEN is %x\n", PWMCFG->TMR1SYNCIEN);

    /* Init TIMER1 */
    TIMER_Init(TIMER1, 1);

    /* Set TIMER1 generate SYNC to PWM when count down to 0 */
    TIMER_EnablePWMSync(TIMER1);

    /* Open Global INT for TIMER1 */
    NVIC_EnableIRQ(TIMER1_IRQn);

    /* Enable Timer */
    TIMER_Enable(TIMER1);

    COMP_SetDACRampDelay(DAC0, 200);
    COMP_SetDACRampDecrement(DAC0, 1);
    COMP_EnableDACRamp(DAC0);

    /* Set DPGAP as input to comparator, set Too High threshold is 3000mV,
filter window is 200ns */
    COMP_Init(COMP_H, COMP_FROM_ANA_IN0, 3000, 200);

    printf("COMP->DAC0CODE is %d\n", COMP->DAC0CODE);

    COMP_SetDACRampReloadValue(DAC0, (COMP->DAC0CODE << 6U));

    printf("COMP->RAMP0MAX is %d\n", COMP->RAMP0MAX);

    while (1)
    {

    }
```

```
}


void TIMER1_IRQHandler()
{
    /* Clear the INT */
    TIMER_ClearInt(TIMER1);
}
```

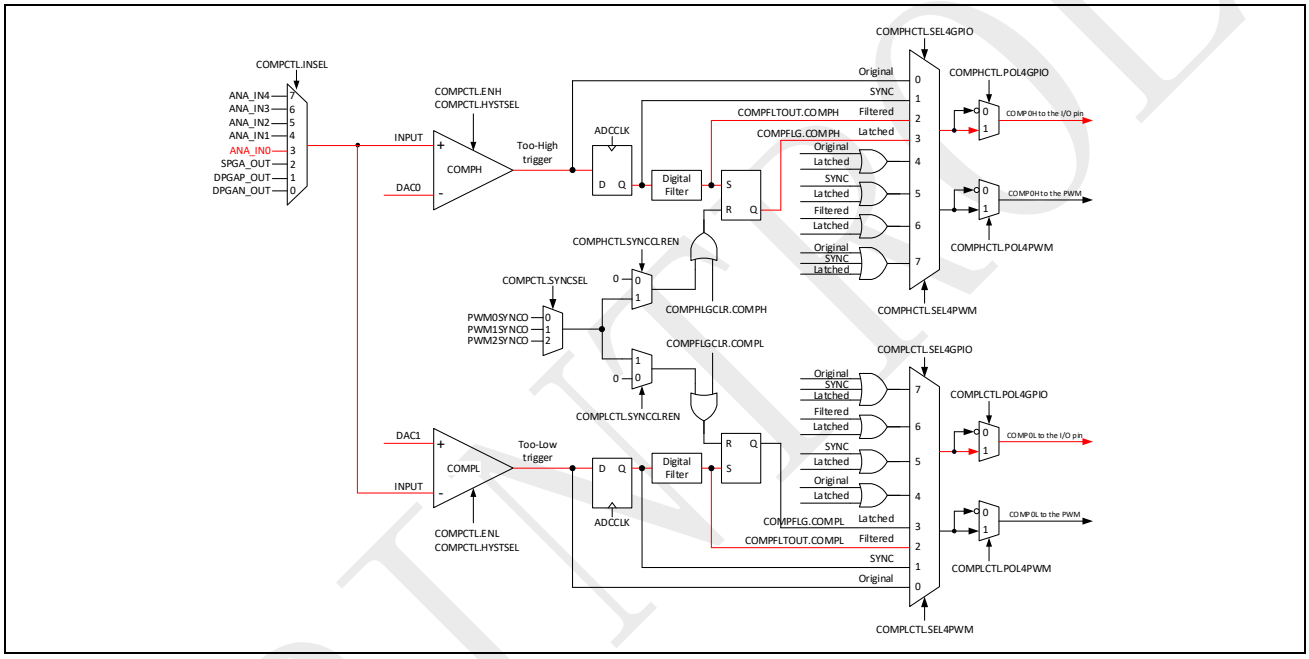
2 COMP 实例

2.1 COMP 示意图

COMP 最常规的用法，是将输入电压与 DAC 比较，从而产生 COMPH 或 COMPL 事件，如图 2-1 所示。

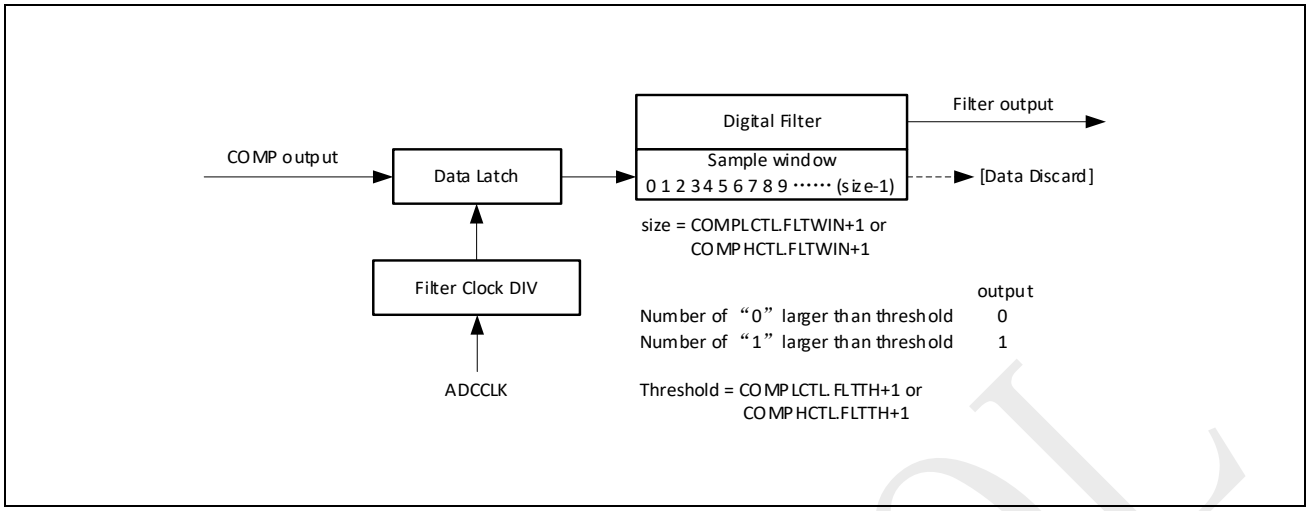
COMP 输入源有 GPIO，SPGA，DPGA，参考电压为 DAC。其输出事件作用于 PWM 模块（产生对应的同步，封锁，TZ 中断，ADC 采样，PWM 中断等事件），其详细信号流图可参考《PWM 使用指南》。

图 2-1：COMP 示意图



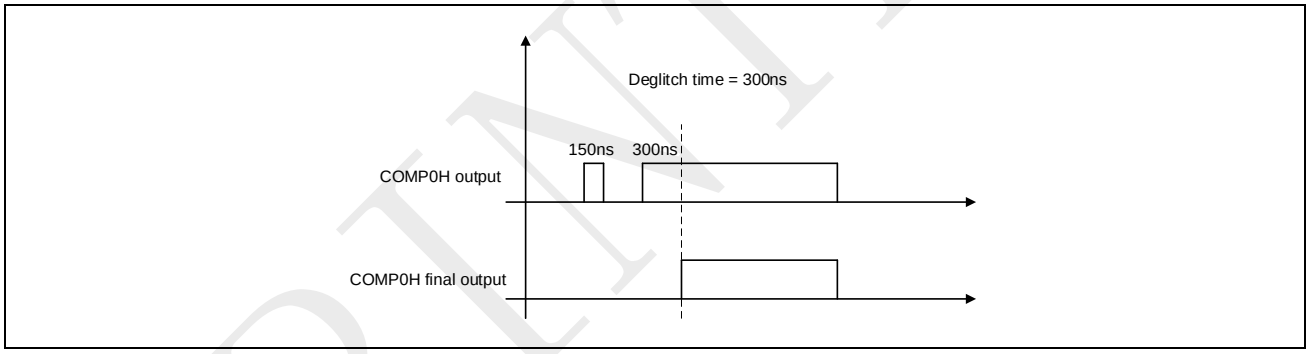
此瞬时响应会误触发 COMP 的输出，并且将 PWM 关断，造成不必要的停机，因此在设计上，提供了输出数字滤波器。其基本原理如下图 2-2 所示，COMP 的输出首先送到锁存器中，时钟来源于 ADCCLK，并且最大可以分频到 1024 倍。然后设置采样窗口的大小和阈值的大小，范围可以从 0~31 中选择，并且要保证阈值要大于采样窗口的一半，如果采样窗口内采到的 COMP 输出 0 的个数大于阈值，则数字滤波器输出为 0，如果采样窗口内的 COMP 输出 1 的个数大于阈值，则数字滤波器输出为 1，如果采用窗口内的 COMP 输出 0 和 1 个数都没有超过阈值，则数字滤波器输出不变。

图 2-2：数字滤波器框图



请执行 COMP 初始化前，务必执行 Clock 的初始化函数，如此才能作正确的配置相关信息。加入数字滤波器之后，COMP 输出如图 2-3 所示，假设设定阀值是 300ns，不足 300ns 的信号被过滤。

图 2-3：数字滤波器模拟框图



以下以一个例子通过 COMP 比较 GPIO0 和 DAC0, DAC1 的电压，并将 COMP 信号输出到 GPIO6，GPIO7 的操作如下。

Example Code

```
int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);
    printf("Enter the test\n");

    PIN_SetChannel(PIN_GPIO0, PIN_GPIO0_ANA_IN0);
```

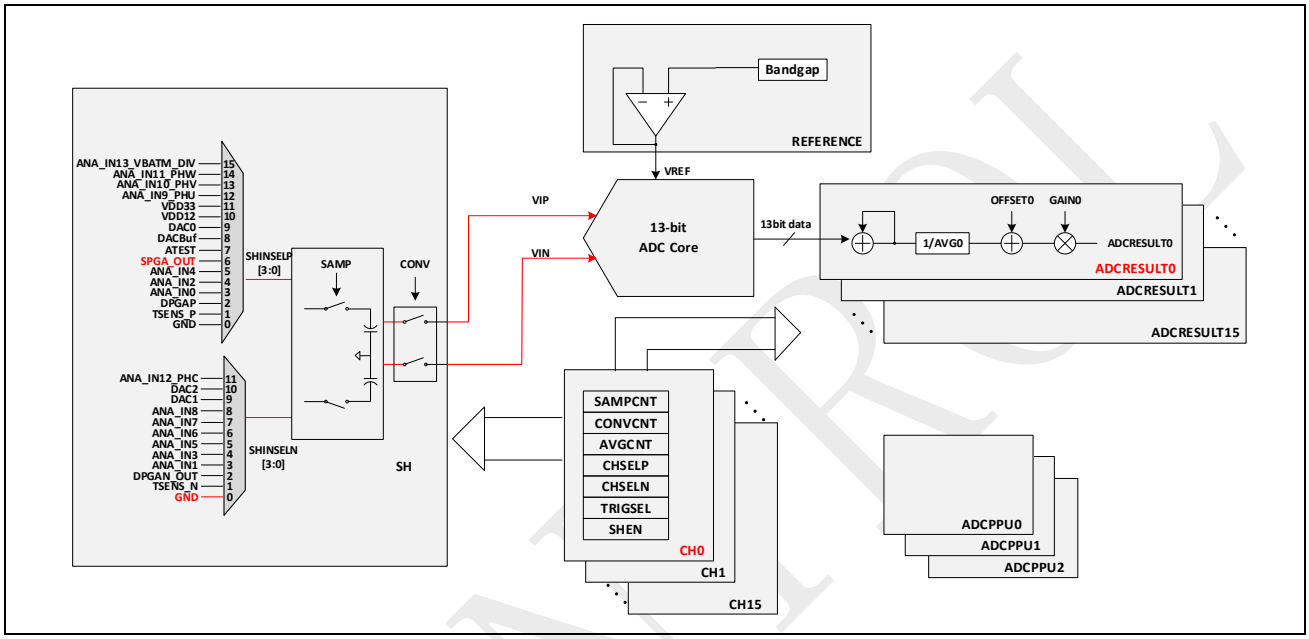
```
    /* Set ANA_IN0 as input to comparator, set Too High threshold is 2500mV,
filter window is 200ns */
    COMP_Init(COMP_H, COMP_FROM_ANA_IN0, 2500, 200);

    /* Set ANA_IN0 as input to comparator, set Too Low threshold is 1900mV,
filter window is 200ns */
    COMP_Init(COMP_L, COMP_FROM_ANA_IN0, 1900, 200);
    PIN_SetChannel(PIN_GPIO6, PIN_GPIO6_COMP_MON2);
    PIN_SetComparatorMonitorSource(PIN_COMP_MONITOR_CH2, PIN_MONITOR_COMP_L);
    COMP_SetOutputPolarityForGPIO(COMP_L, HIGH);

    PIN_SetChannel(PIN_GPIO7, PIN_GPIO7_COMP_MON3);
    PIN_SetComparatorMonitorSource(PIN_COMP_MONITOR_CH3, PIN_MONITOR_COMP_H);
    COMP_SetOutputPolarityForGPIO(COMP_H, HIGH);

    while (1)
    {
        /* Get the comparator status */
        if (COMP_GetFilterOutputFlag(COMP_H) != 0)
        {
            printf("ANA_IN0 is higher than 2500mV\n");
        }
        else if (COMP_GetFilterOutputFlag(COMP_L) != 0)
        {
            printf("ANA_IN0 is lower than 1900mV\n");
        }

        /* Clear latched filter status */
        COMP_ClearFilterOutputFlag(COMP_L | COMP_H);

        Delay_Ms(500);
    }
}
```

3 PGA 实例

3.1 SPGA 送入 ADC 采样

使用 ADC 对 SPGA 采样，其链接图如图 3-1 所示。

图 3-1：ADC SPGA 采样



以下例子演示使用 ADC 对 SPGA 电压进行采样，将 SPGA 电压送给 ADC CH0 进行采样，默认采样时间，转换时间均设定为 200ns。但在实际的工程中采样时间会受到外部负载的影响，其具体的设置数值，可参考《ADC 建立时间计算方法使用指南》，而转换时间则不会受到外部负载影响，通常保持 SDK 中设定的数值即可。

在实际的使用中 SPGA 输入电压必须符合表 3-1。

表 3-1：给定增益下 SPGA 输入输出范围

SPGACTL.GAIN	GAIN	Input range (V)			Output range (V)
0	1	0.3	~	2	0.3~2
1	2	0.15	~	1.5	0.3~3
2	4	0.075	~	0.75	0.3~3
3	8	0.0375	~	0.375	0.3~3
4	16	0.01875	~	0.1875	0.3~3
5	32	0.009375	~	0.09375	0.3~3
6	48	0.00625	~	0.0625	0.3~3
7	64	0.0046875	~	0.046875	0.3~3

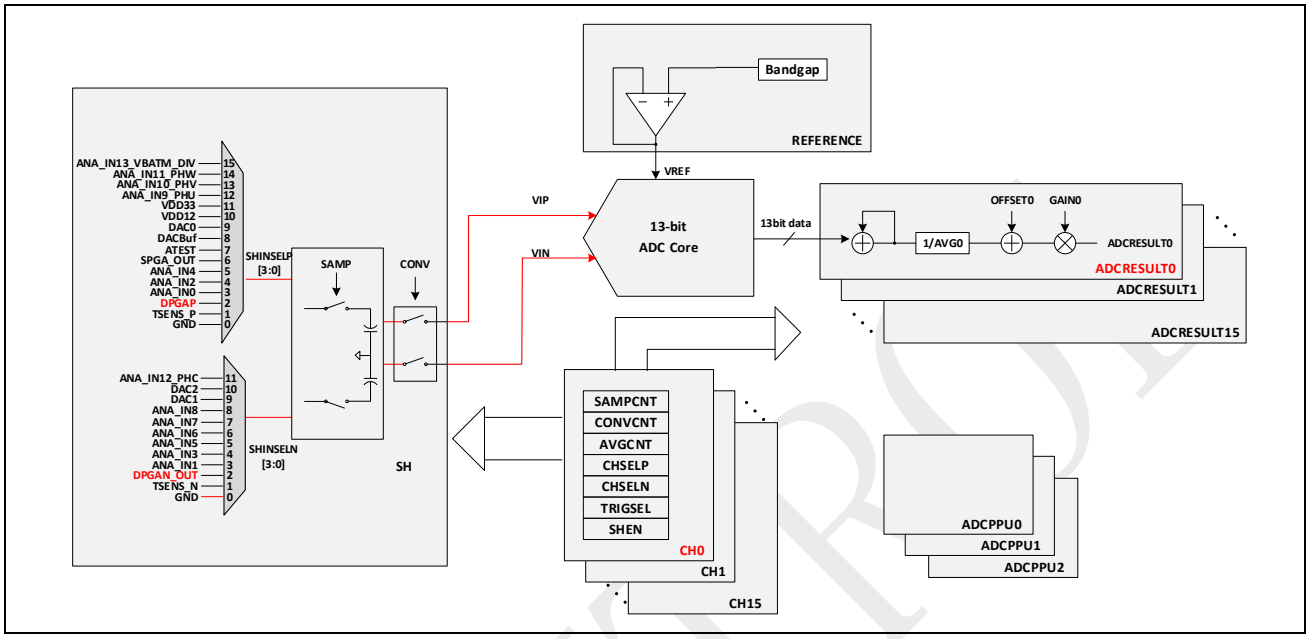
Example Code

```
int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

    /*
    * set the GPIO as ADC.
    */
    PIN_SetChannel(PIN_GPIO0, PIN_GPIO0_ANA_IN0);

    /*
    * Set PGA in  single-ended mode.
    */
    PGA_InitSPGA(SPGA_FROM_ANA_IN0, SPGA_GAIN_1X);

    /*ADC Init*/
    ADC_EasyInit1(ADC, ADC_CH0, ADC_IN_SPGA_OUT, ADC_SOC_TRIGGER_FROM_SOFTWARE);

    /* Set Average Times */
    ADC_SetChannelResultAverageCount(ADC, ADC_CH0, ADC_AVERAGE_COUNT_16);

    while (1)
    {
        /* Use software to trigger ADC SOC0 start to work */
        ADC_ForceChannelSOC(ADC, ADC_CH0);

        /* Wait until ADC conversion finished (Interrupt flag was set) */
        while (ADC_GetChannelIntFlag(ADC, ADC_CH0) == 0);

        /* Get result */
        i32VSP = ADC_GetChannelResult(ADC, ADC_CH0);

        i32VSP = ABS(i32VSP);

        /* Clear ADC SOC0 INT flag */
        ADC_ClearChannelInt(ADC, ADC_CH0);

        printf("ADC Single End Result = %d\n", i32VSP);
        printf("ADC Single End Voltage %dmv\n", ValueToVoltage(i32VSP));

        Delay_Ms(500);
    }
}
```

3.2 DPGA 送入 ADC 采样

使用 ADC 对 DPGA 采样，其链接图如图 3-2 所示。

图 3-2：ADC DPGA 采样



以下例子演示使用 ADC 对 DPGA 电压进行采样，将 DPGA 电压送给 ADC CH0 进行采样，默认采样时间，转换时间均设定为 200ns。但在实际的工程中采样时间会受到外部负载的影响，其具体的设置数值，可参考《ADC 建立时间计算方法使用指南》，而转换时间则不会受到外部负载影响，通常保持 SDK 中设定的数值即可。

在实际的使用中 DPGA 输入电压必须符合表 3-2。

表 3-2：给定增益下 DPGA 输入输出范围

DPGACTL.GAIN	$Gain_{DPGA}$	Input range (V)	Output range (V)
0	2	-1.35~1.35	0.3~3
1	4	-0.675~+0.675	0.3~3
2	8	-0.337~+0.337	0.3~3
3	16	-0.168~+0.168	0.3~3
4	24	-0.112~+0.112	0.3~3
5	32	-0.084~+0.084	0.3~3
6	48	-0.056~+0.056	0.3~3
7	64	-0.042~+0.042	0.3~3

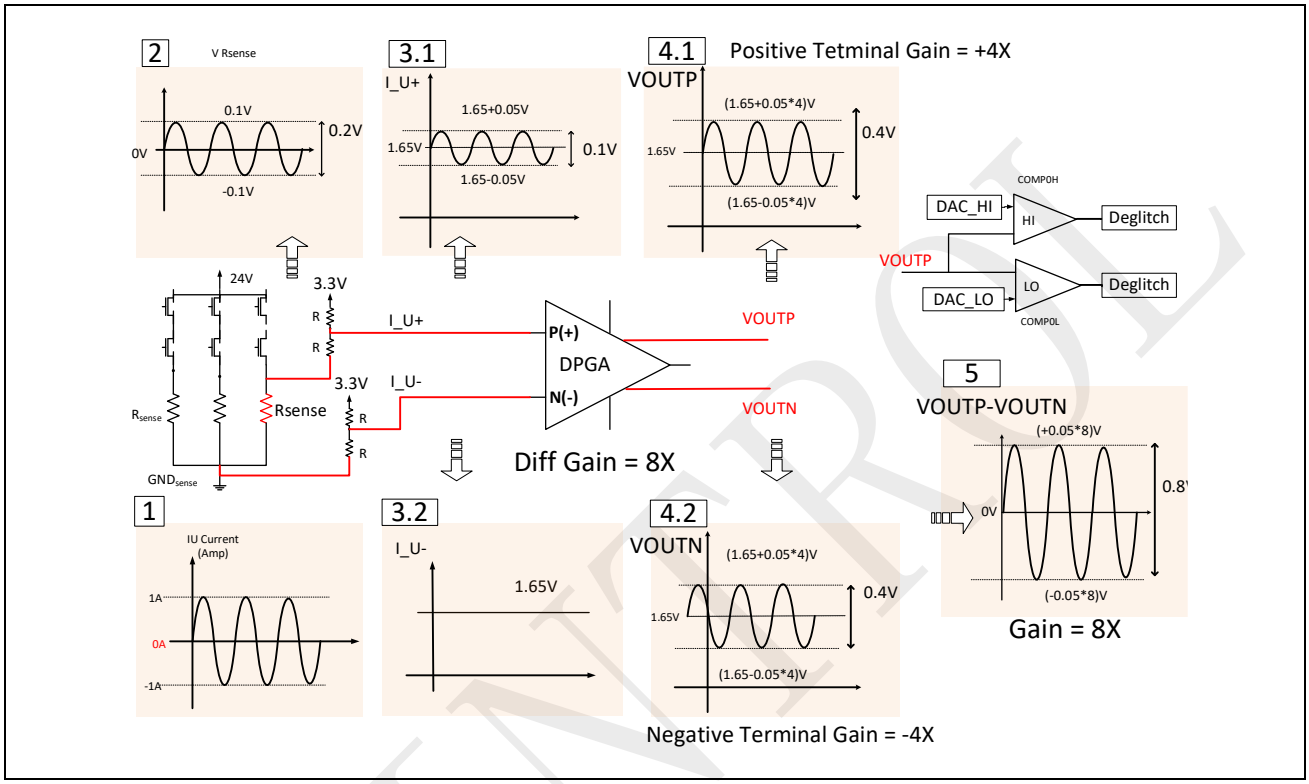
Example Code

```
int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

    /*
    * Set PGA in  differential-ended mode.
    *
    */
    PGA_InitDPGA(DPGA_GAIN_2X);

    /*ADC Init*/
    ADC_EasyInit2(ADC, ADC_CH0, ADC_IN_DPGAP_OUT, ADC_IN_DPGAN_OUT,
ADC_SOC_TRIGGER_FROM_SOFTWARE);

    /* Set Average Times */
    ADC_SetChannelResultAverageCount(ADC, ADC_CH0, ADC_AVERAGE_COUNT_16);
    while (1)
    {
        /* Use software to trigger ADC SOC0 start to work */
        ADC_ForceChannelSOC(ADC, ADC_CH0);

        /* Wait until ADC conversion finished (Interrupt flag was set) */
        while (ADC_GetChannelIntFlag(ADC, ADC_CH0) == 0);

        /* Get result */
        i32VSP = ADC_GetChannelResult(ADC, ADC_CH0);

        /* Clear ADC SOC0 INT flag */
        ADC_ClearChannelInt(ADC, ADC_CH0);

        printf("ADC differential Result = %d\n", i32VSP);
        printf("ADC differential Voltage %dmv\n", ValueToVoltage(i32VSP));

        Delay_Ms(500);
    }
}
```

3.3 使用 DPGA 进行过电流保护

在实际使用场景中，通常会利用 DPGA 放大 R_{sense} 上的电流，此时必须将信号的正端与负端均偏置到 VDD/2 的地方，图 3-3 所示。

图 3-3：DPGA 放大信号示意图



以下以一个例子分析内部电压：

- 假设 R_{sense} 为 0.1 欧姆，U 相电流幅值为 1A；
- 设定的差分放大增益是 8X；
- R 建议为 10k 欧姆；
- 根据电路叠加原理，DPGA 正端输入电压为 $\frac{3.3}{2}+\frac{0.1*\sin(wt)}{2}$ 负端输入电压为$\frac{3.3}{2}$；
- 最终输出信号$VOUTN$和$VOUTP$如下：

$$VCM=\frac{DVDD33}{2}=1.65$$

$$VOUTN=VCM-VIN*\frac{G}{2}=1.65-\frac{0.1*\ sin(wt)}{2}*\frac{8}{2}=1.65-\ 0.2*\ sin(wt)$$

$$VOUTP=VCM+VIN*\frac{G}{2}=1.65+\frac{0.1*\ sin(wt)}{2}*\frac{8}{2}=1.65+\ 0.2*\ sin(wt)$$

式中 VCM 为输出共模电压，芯片内部固定为$\frac{DVDD33}{2}$，VIN 为 DPGA 输入端的差模电压

$\frac{0.1*\sin(wt)}{2}$，G 为 DPGA 的放大倍数 8X。

- COMP 选择$VOUTP$作为输入，因此其偏移为 1.65V，幅值为 0.2V；
- 假设 U 相过电流幅值 2A，是额定相电流幅值的 2 倍，因此 DAC 的参考电平偏移为 1.65V，幅值为 2*0.2V；

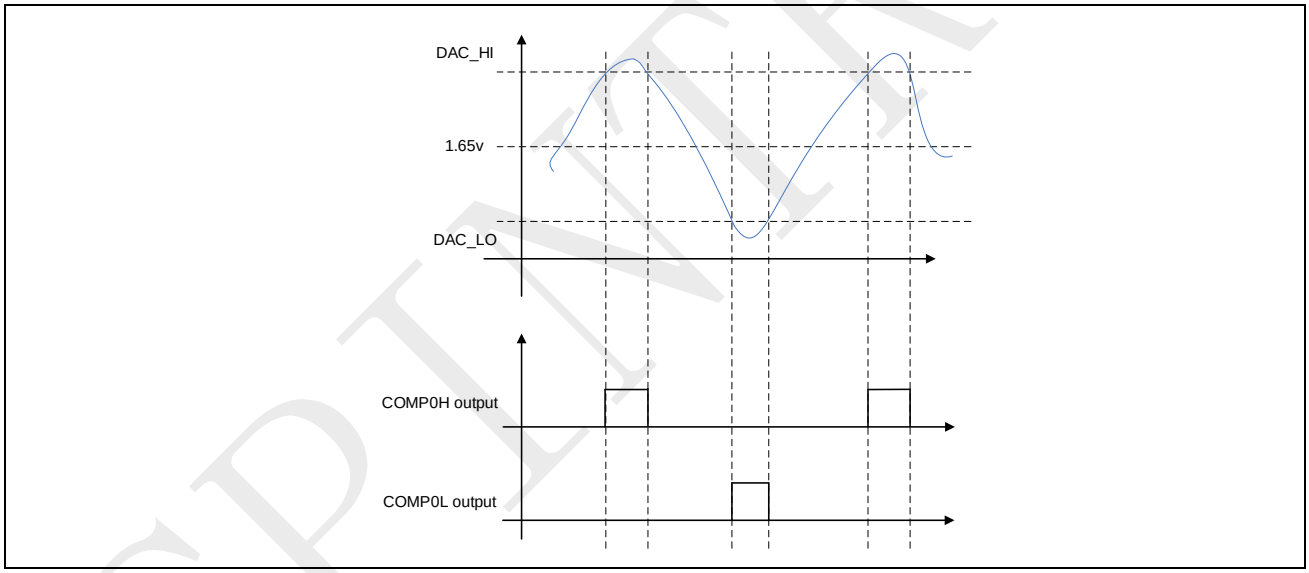
$$DAC_{HI} = 1.65 + 2 * 0.2 = 2.05V$$
$$DAC_{LO} = 1.65 - 2 * 0.2 = 1.25V$$

- 根据以下公式，可以决定DAC_{HI}与DAC_{LO}的设定数值；

$$DAC_{HI} = \frac{DAC\_HI\_CODE(10bit)}{1024} \times 3.3V$$
$$DAC_{LO} = \frac{DAC\_LO\_CODE(10bit)}{1024} \times 3.3V$$

- DAC_HI_CODE（寄存器数值）须设定为 636，DAC_LO_CODE（寄存器数值）须设定为 387。为了使用方便，Spintrol 提供之 COMP_Init（）函数可直接输入 mV 值进行设定。
- 根据上一小节的设计，当输入电流过大或过小时，都会触发 COMP 输出；

图 3-4：COMP 输出演示

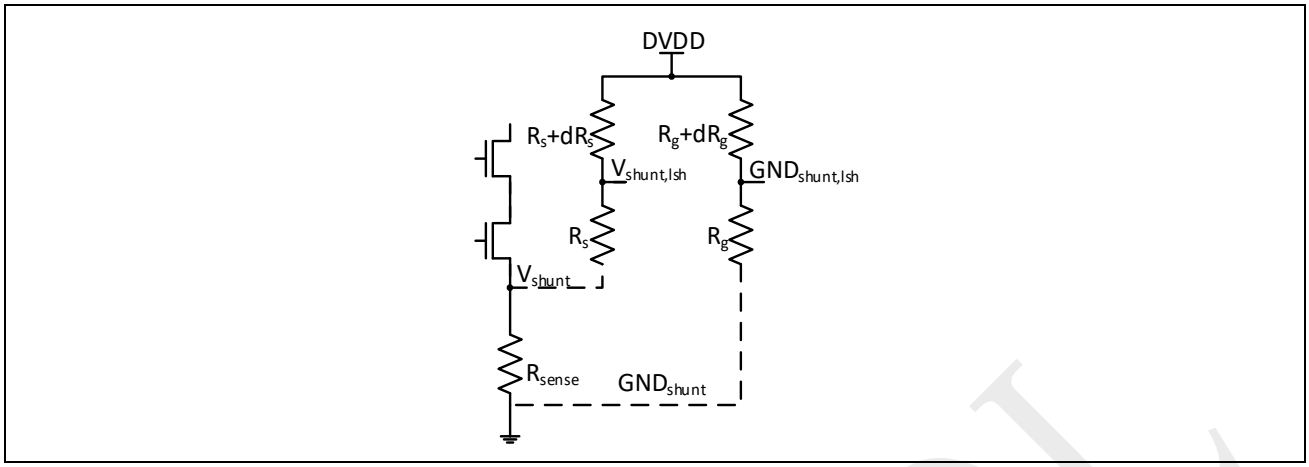


PWM 控制 H 桥电路，在电机控制系统或是电源控制系统是非常常见的技术，但是在 PWM 开关的瞬间，往往会对电流回授电路有较大的干扰，或是开关的瞬间常有较大的电流瞬时（Current Transient）。解决方法参见章节 2 中 COMP 滤波器。

3.3.1 电平移位器校准

图 3-3 每个电阻电平移位器由两个电阻组成。两个电阻阻值应相等，以便使信号向上移位 AVDD/2。这两个电阻通常会有不匹配，这种不匹配需要校准，以便确定 Rsense 两端的电压。如图 3-5 所示。

图 3-5：电平移位器校准



为简单起见，但不降低讨论的一般性， Vshunt 和 GNDshunt 电压，理想中应该向上移位 AVDD/2，但是由于电阻不匹配，实际移位值会所差别。根据下面的表达式，可以计算出移位后的电压值：

$$V_{shunt,lsh} = V_{ddx}\frac{R_s}{2\cdot R_s + \delta R_s} + V_{shunt}\frac{R_s + \delta R_s}{2\cdot R_s + \delta R_s} = \frac{V_{ddx}}{2}\left[1 - \frac{\delta_s}{2}\right] + \frac{V_{shunt}}{2}\left[1 + \frac{\delta_s}{2}\right]$$

$$GND_{shunt,lsh} = V_{ddx}\frac{R_g}{2\cdot R_g + \delta R_g} + GND_{shunt}\frac{R_g + \delta R_g}{2\cdot R_g + \delta R_g} = \frac{V_{ddx}}{2}\left[1 - \frac{\delta_g}{2}\right] + \frac{GND_{shunt}}{2}\left[1 + \frac{\delta_g}{2}\right]$$

在这里，Rs/Rs=δs，Rg/Rg=δg。因此，移位后的差分输出电压为：

$$V_{ind} = V_{shunt,lsh} - GND_{shunt,lsh} = \frac{V_{ddx}}{2}\cdot\frac{\delta_g - \delta_s}{2} + \frac{V_{shunt} - GND_{shunt}}{2} + \frac{V_{shunt}}{2}\cdot\frac{\delta_s}{2}$$

这里，忽略了 GNDshuntδg/4，因为 GNDshunt 和δg 都是小量。我们发现失配误差可以分为增益误差（Vshunt 和δs /2 乘积）和偏移误差（Vshunt=0 时的 Vind）。增益误差项相当于 Rsense 电阻的变化。这两个项都可以在当电机不工作时通过测量电压来确定，具体步骤如下：

- 测量 AVDD。可以将 AVDD 通过 ADC 多路选择器送到 ADC 测量。
- 当电机不工作时，没有电流通过 Rsense，也就是 Vshunt=0，测量 Vshunt,lsh。通过上面 Vshunt,lsh 的表达式可以计算出δs。
- 当电机不工作时，没有电流通过 Rsense，测量 Vind。根据 Vind 的表达式可以计算出$\delta_g - \delta_s$。

Example Code

```
void PWMx_Set_TZ_Event(PWM_REGS *PWMx)
{
    /* Affected by results monitored from COMP0 */
    PWM_EnableDCAHTripEvent(PWMx, DC_TRIP_COMP0H);
    PWM_EnableDCAHTripEvent(PWMx, DC_TRIP_COMP0L);

    /*DCAEVT0 event comes from DCAL=don't care, DCAH=high*/
    PWM_SetRawDCAEVT0(PWMx, DCH_HIGH_DCL_X);

    /* Set the filter, Filter trigger conditions is TBCNT=0*/
    PWM_SetDCFilter(PWMx, DCF_FROM_RAW_DCAEVT0, DCF_ALIGN_ON_ZERO);

    /* Enable PWM DC filter blank function */
    PWM_EnableDCFilterBlank(PWMx);
```

```
    /* Set blank window size as 100 TBCLK and offset as 50 TBCLK */
    PWM_SetDCFilterBlankWindow(PWMx, PWM_BlankWIN_Size, PWM_Blank_Offset);

    /* USe the filtered to deal the data from DCAEVT0 */
    PWM_SetDCAEVT0(PWMx, DCEVT_FILTERED);

    /* Set the DCAEVT0 as OneShot mode */
    PWM_SetOneShotTripEvent(PWMx, TRIP_EVENT_DCAEVT, TRIP_OUTPUT_LATCH);

    /* Set output to tri-state upon OST trip event */
    PWM_SetCHAOutputWhenTrip(PWMx, TZU_TRIP_AS_LOW |
                           TZD_TRIP_AS_LOW |
                           DCEVT0U_TRIP_DO_NOTHING |
                           DCEVT0D_TRIP_DO_NOTHING |
                           DCEVT1U_TRIP_DO_NOTHING |
                           DCEVT1D_TRIP_DO_NOTHING);

    PWM_SetCHBOutputWhenTrip(PWMx, TZU_TRIP_AS_LOW |
                           TZD_TRIP_AS_LOW |
                           DCEVT0U_TRIP_DO_NOTHING |
                           DCEVT0D_TRIP_DO_NOTHING |
                           DCEVT1U_TRIP_DO_NOTHING |
                           DCEVT1D_TRIP_DO_NOTHING);

    PWM_EnableTripInt(PWMx, TRIP_INT_OST);
}


int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("Enter the test\n");

    /*
    * Set PGA in  differential-ended mode.
    *
    */
    PGA_InitDPGA(DPGA_GAIN_8X);

    /*
     * 1. Initialize comparator with 300ns deglitch filtering window.
     * 2. Set DAC voltage, COMP_HI's negative port is 2050mV,
     * COMP_LO's positive port is 1250mV
     */
    COMP_Init(COMP_H, COMP_FROM_DPGAP_OUT, SampleRgisterNVol + VolOffsetmV, \
            COMP_FilterWIN);
    COMP_Init(COMP_L, COMP_FROM_DPGAP_OUT, SampleRgisterNVol - VolOffsetmV, \
            COMP_FilterWIN);

    /* Select the channel A/B output of PWM1 respectively */
    PIN_SetChannel(PIN_GPIO12, PIN_GPIO12_PWM1A);
    PIN_SetChannel(PIN_GPIO13, PIN_GPIO13_PWM1B);
```

```
    PWM_InitComplementaryPairChannel(PWM1, PWM_FREQ, PWM_DB_NS);

    /* Set PWM1A output 25% duty waveform */
    u32PWMPeriod = PWMPeriod(PWM_FREQ);
    PWM_SetCMPA(PWM1, (u32PWMPeriod * 3) / 4);

    /* Start counting */
    PWM_RunCounter(PWM1);

    /* Set the DC sub-module for PWM1 */
    PWMx_Set_TZ_Event(PWM1);

    NVIC_EnableIRQ(PWM1TZ_IRQn);

    while (1)
    {
        Delay_Ms(1000);

        /* Restore the output of wave in oneshot mode*/
        PWM_ClearTripInt(PWM1, TRIP_INT_OST);
    }
}

void PWM1TZ_IRQHandler(void)
{
    if (PWM_GetOneShotTripEventFlag(PWM1, TRIP_EVENT_DCAEVT))
    {
        printf("one-shot trip event occurred\n");

        PWM_ClearOneShotTripEventFlag(PWM1, TRIP_EVENT_DCAEVT);
    }
    PWM_ClearTripInt(PWM1, TRIP_INT_GLOBAL);
}
```

除了使用 PGA 进行过电流保护外，SPC1169 内部集成了针对三相电流桥上下桥臂外部 Mos 管的过流保护电路，相比 PGA 的实现方式，其代码配置大大简化，详细说明见《预驱使用指南》。